① Traverse Operation

Traversal operation in array or simply traversing an array means, Ascending and printing each elements of an array.
Traverse: print all the array elements one by one

Algorithm: Step1 : Start
Step2 : [initialize Counter variable] Set i=LB
Step3 : Repeat for i=LB to UB
Step4 : Apply process to arr[i]
Step5 : [End of Loop]
Step6 : Stop.

② Insertion Operation

Insert operation is to insert one or more data elements into an array. Based on the requirement a new element can be added at the beginning, end, or any given index of array.
Insertion - Adds an element at the given index.

Algorithm: Step1: Start
Step2 : [Reset size of array] Set size = Size+1
Step3 : [initilize Counter variable] Set i = size-i
Step4 : Repeat step5 and step6 for i = size-1 to i >= pos-1
Step5 : [Move ith element forword] set arr[i+1] = arr[i]
Step6 : [Decrease Counter] Set i = i-1
Step7 : [End of Step 4 loop]
Step8 : [Insert element] Set arr[pos-1] = x
Step9 : Stop.

③ Deletion Operation

Deletion refers to removing an existing element from the array and re-organizing all elements of an array.

Algorithm : Step 1 : Start
Step 2 : Set J = K.
Step 3 : Repeat Step 4 and 5 while J < N
Step 4 : Set LA[J] = LA[J+1]
Step 5 : Set J = J+1
Step 6 : Set N = N-1.
Step 7 : Stop.

④ Search Operation

You can perform a search for an array element based on its values or its index.
Search : Searches an element using the given index or by the value.

Algorithm : Step 1 : Start
Step 2 : Stet J = 0
Step 3 : Repeat Step 4 and 5 while J < N.
Step 4 : If LA[J] is equal ITEM then go to Step 6.
Step 5 : Ste Set J = J+1
Step 6 : Print J, ITEM.
Step 7 : Stop.

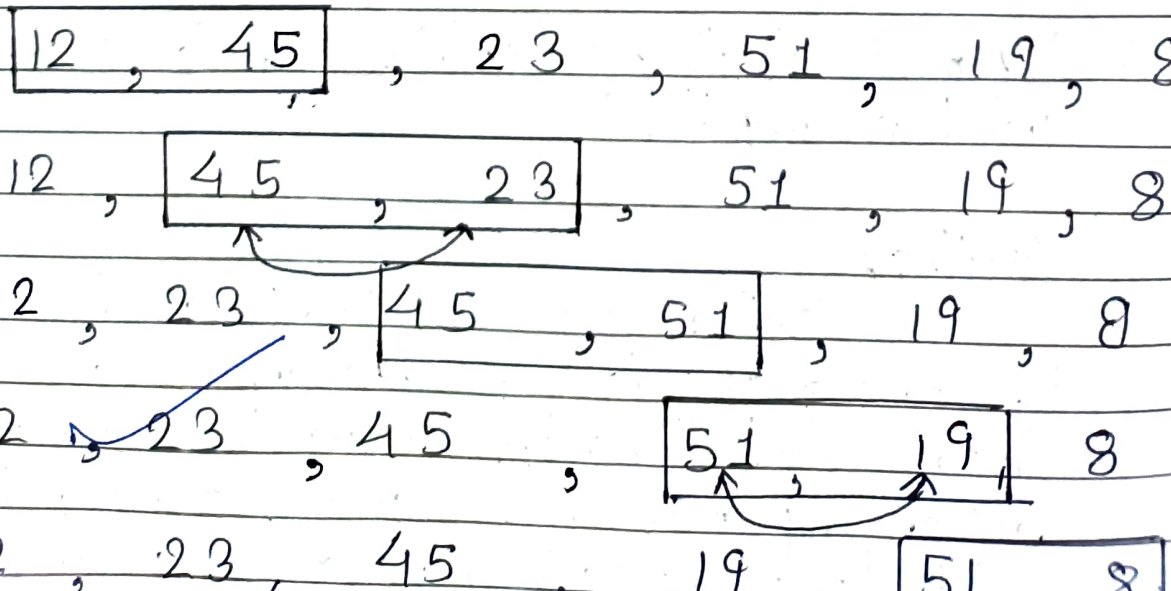
Bubble Sort:

Bubble sort works on the repeatedly swapping of adjacent elements until they are not in the intended order. It is called bubble sort because the movement of array elements is just like the movement of air bubbles in the water. Bubble in the water rise up to the surface; similarly, the array elements in bubble sort move to the end in each iteration.

Algorithm - step1 : begin BubbleSort (arr)
Step2 : for all array elements
Step3 : if arr [i] > arr [i+1]
step4 : Swap (arr [i], arr [i+1])
Step5 : End if
Step6 : end for loop
Step7 : return arr
Step8 : End BubbleSort.

Working of Bubble Sort:

I.



Ques 5. Write the Algorithm of Insertion and Selection Sort.

→ • Algorithm of Insertion Sort:

Step 1: If the element is the first element, assume that it is already sorted. Return 1.
Step 2: Pick the next element, and store it seperately in a key.
Step 3: Now compare the key with all elements in the sorted array.
Step 4: If the element in the sorted array is smaller than the current element, then move to the next element. Else, shift greater elements in the array towards the right.
Step 5: Insert the value
Step 6: Repeat until the array is sorted.

• Algorithm of Selection Sort:

Step 1: Selection Sort (arr, n)
Step 2: Call Smallest (arr, i, n, pos)
Step 3: Swap arr[i] with arr[pos]
Step 4: Repeat Step 2 & 3 for i=0 to n-1
Step 5: End the Loop
Step 6: Exit.
Step 7: Smallest (arr, i, n, pos)
Step 8: [initialize] Set small = arr[i]

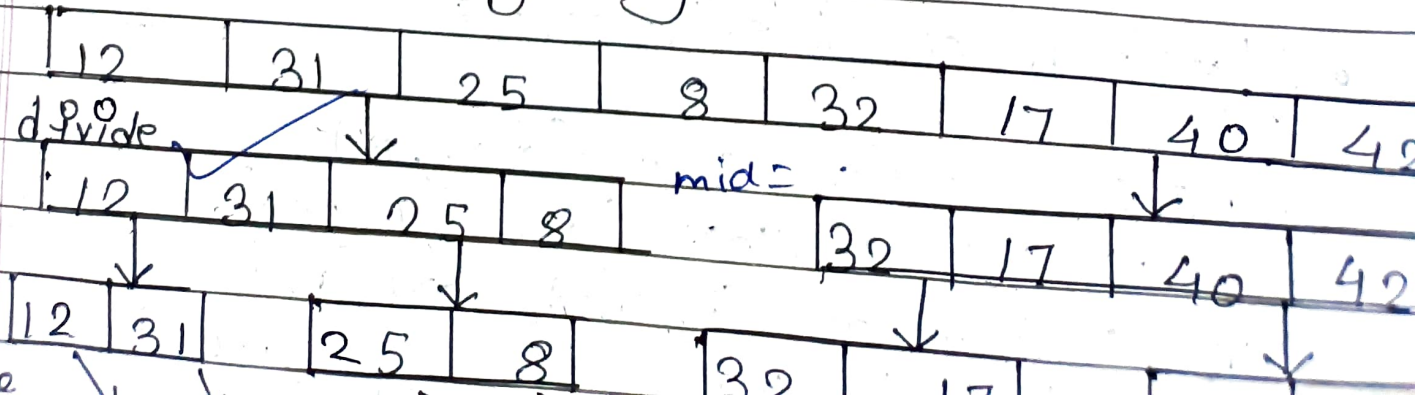
Step 9 : [initialize] Set pos = i
Step 10 : Step 3 : Repeat for j = i+1 to n
Step 11 : If (small > arr [j])
Step 12 : Set small = arr [j]
Step 13 : Set pos = j
Step 14 : End of if
Step 15 : End of loop
Step 16 : Return pos.

Ques. What is merge Sort. Explain working with example.

→ Merge Sort is similar to quick Sort algorithm as it uses the divided and Conquer approach to so the elements. It is one of the most popular efficient Sorting algorithm. It divides the given list into two equal halves, calls it for the two halves and then merges the two sorted halves. We have to define the merge () function to perform the merging.

- Working of Merge Sort Algorithm:

Let the element of Array are:



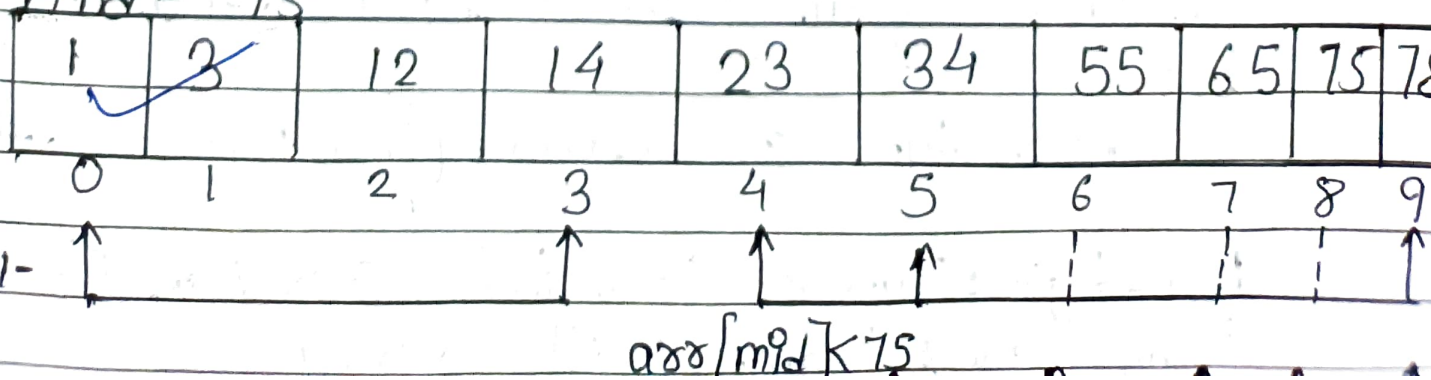
Algorithm of the Linear Search:-

Step 1 - Set i to 0
Step 2 - if i>n then go to Step 7.
Step 3 - If Arr[i] = a then go to Step 6
Step 4 - Set i to i+1
Step 5 - Go to Step 2.
Step 6 - Print Element is found at Index i and go to Step 8
Step 7 - Print Element is not found.
Step 8 - Exit.

(2) Binary Search:

Binary Search algorithm falls under the categories of interval Search algorithm. This algorithm is much more efficient compared to linear search. Binary search only works on sorted data structure. This algorithm repeatedly target the center of the sorted data structure & divide the Search space into half till the match is found. The time complexity of binary search algorithm is O(log n)

Find - '75'



Algorithm of Binary Search:

1. Take input array, left, right, & x.
2. Start loop - while (left greater than or equal to right)
   - mid = left + (right - left) / 2
   - if (arr[mid] == x) then
     - return m
   - else if (arr[mid] less than x) then
     - left = m + 1
   - else
     - right = mid - 1
3. End Loop
4. Return -1

Q. Write a note on Multidimensional Array.

```
int data = queue[front];
front = front+1;
return data;
}
```

③ peek(): Gets the element at the front of the queue without removing it.

④ isFull(): Checks if the queue is full or not.
⑤ isEmpty: Checks if the queue is empty or not.

Ques2. Write the Algorithm of isempty, isfull, peek.

→ ① isfull() algorithm:

Step1: Check if rear = Maxsize - 1
Step2: if they are equal, return "Queue is not full".

② isempty() Algorithm:

Step1: check if the rear and front an pointing to null memory space ie -1

Step2: If they an pointing to -1, return "queue is empty".

Step3: If they are not equal, return "Queue is not empty.

③ Peek() Algorithm:

Step1: Check If the queue is empty

Step2: If they queue is empty, return. "Queue is empty".

Step3: If the queue is not empty, access the data where the front pointer is pointing.